Center for Embedded Computer Systems
University of California, Irvine

Cross-Layer Interactions of Error Control Schemes in Mobile Multimedia Systems

Kyoungwoo Lee, Aviral Shrivastava, Minyoung Kim, Nikil Dutt, and Nalini Venkatasubramanian

Technical Report TR-08-09
July 28, 2008

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA
(949) 824-8059

{kyoungwl, minyounk, dutt, nalini}@ics.uci.edu, Aviral.Shrivastava@asu.edu
http://www.cecs.uci.edu/

Cross-Layer Interactions of Error Control Schemes in Mobile Multimedia Systems

Kyoungwoo Lee, Aviral Shrivastava, Minyoung Kim, Nikil Dutt, and Nalini Venkatasubramanian

Technical Report TR-08-09
July 28, 2008

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA
(949) 824-8059

{kyoungwl, minyounk, dutt, nalini}@ics.uci.edu, Aviral.Shrivastava@asu.edu
http://www.cecs.uci.edu

Abstract

Soft errors are threatening the system reliability in mobile devices but traditional hardware protection techniques incur significant overheads in terms of power and performance. Thus, incorporating reliability in resource-limited mobile devices poses significant challenges. This paper discusses a cooperative method that exploits existing error control schemes at an application layer to mitigate the impact of hardware defects such as soft errors for mobile multimedia systems. So we study heterogeneous specifics about two different errors at two different abstraction layers, and present a cooperative cross-layer approach to obtain low-cost reliability at the minimal degradation of QoS. In particular, we propose a cooperative approach to combat soft errors at data caches by using dual schemes – a Drop and Forward Recovery and an error-resilient video encoding – driven by intelligent middleware schemes. Experimental evaluation demonstrates that our cooperative error-aware method for a video encoding with different video streams improves performance by 60% and the energy consumption by 58% with even better reliability at the cost of 3% quality degradation on average, as compared to an ECC-based hardware protection technique. Combining intelligent schemes to select a recovery mechanism can guide system designers for trading off multiple constraints such as performance, power, reliability, and QoS.

Contents

List of Figures

List of Tables

Cross-Layer Interactions of Error Control Schemes in Mobile Multimedia Systems

K. Lee[1], A. Shrivastava[2], M. Kim[1], N. Dutt[1], and N. Venkatasubramanian[1]

[1]Department of Computer Science
School of Information and Computer Sciences
University of California, Irvine, CA 92697, USA

[2]Department of Computer Science and Engineering
School of Computing and Informatics
Arizona State University, Tempe, AZ 85281, USA

July 28, 2008

Abstract

Soft errors are threatening the system reliability in mobile devices but traditional hardware protection techniques incur significant overheads in terms of power and performance. Thus, incorporating reliability in resource-limited mobile devices poses significant challenges. This paper discusses a cooperative method that exploits existing error control schemes at an application layer to mitigate the impact of hardware defects such as soft errors for mobile multimedia systems. So we study heterogeneous specifics about two different errors at two different abstraction layers, and present a cooperative cross-layer approach to obtain low-cost reliability at the minimal degradation of QoS. In particular, we propose a cooperative approach to combat soft errors at data caches by using dual schemes – a Drop and Forward Recovery and an error-resilient video encoding – driven by intelligent middleware schemes. Experimental evaluation demonstrates that our cooperative error-aware method for a video encoding with different video streams improves performance by 60% and the energy consumption by 58% with even better reliability at the cost of 3% quality degradation on average, as compared to an ECC-based hardware protection technique. Combining intelligent schemes to select a recovery mechanism can guide system designers for trading off multiple constraints such as performance, power, reliability, and QoS.

1 Motivation

With advances in processor and wireless communication technologies, mobile devices such as PDA and smart phones have emerged as a main component for a range of multimedia applications such as video telephony and remote image sensing. Challenges to cope with error-prone transmission over

wireless networks bring out the need for error-resilient techniques, and they must be implemented in an energy-efficient way to prolong the life of battery-powered mobile devices. Note that the main objective of error-resilient techniques, e.g., an error-resilient video encoding, is to recover the erroneous video data due to transmission errors for maintaining the video quality.

Increasing exponentially with each technology generation, soft errors will soon become an everyday concern [11, 38]. Soft errors are transient faults that are caused due to a variety of deep submicron reasons, including sudden voltage drops, signal interference, random noise, etc., but cosmic radiation strike causes more soft errors than all other reasons put together [2]. Soft errors are emerging as a problem in mobile multimedia devices, since these consumer devices – for competitive market reasons – increasingly deploy components manufactured using the latest technology, and operate at low voltages for extended battery life. Both of these factors reduce the threshold charge $Q_{critical}$ for a striking radiation particle to cause a soft error, which in turn greatly affects the reliability. Since memories occupy majority real estate on-chip, memories are most vulnerable to soft errors. Mobile multimedia devices are especially prone to soft errors owing to i) the sheer volume of data processed in multimedia applications, and ii) they are more likely to be used in environments with high soft error rates, e.g., mountain tops and airplanes.

Solutions to reduce soft errors have been proposed at all levels of design hierarchy from hardening devices [3, 29] to error control schemes such as TMR (Triple Modular Redundancy) [27] and ECC (Error Correction Codes) [27]. However, these techniques incur high overheads in terms of power and performance [18, 19]. For example, TMR typically uses three functionally equivalent replicas of a logic circuit and a majority voter, but the overheads of hardware and power for conventional TMR exceed 200% [25]. Also, implementing an ECC-based scheme, the most popular one in memory systems, raises access time by up to 95% [18] and power consumption by up to 22% [26] in the caches. To reduce the overheads, several microarchitectural solutions [17, 19, 24] have been investigated but still incur overheads in terms of power and performance.

On the other hand, an EDC (Error Detection Codes) based technique such as parity codes [27] incurs much less overheads than an ECC-based technique such as a Hamming code [27]. For example, a parity code decreases the access delay by up to 47% and the power consumption by up to 73% compared to a Hamming code (38,32) according to [19]. However, an EDC scheme can only detect an error while an ECC can even correct it. Thus, to make a system tolerant against soft errors, an EDC scheme must be followed by an error recovery technique such as rolling-backward recovery with checkpoints [27]. Checkpoints are made every interval and the system states are saved at the reliable storage. Once an error is detected, the system rolls backward to the last saved checkpoint and re-processes the functionality after a recovery – backward error recovery (BER). However, recovery with checkpoints is inappropriate for real-time applications since they have in general poor predictability of the completion time – a checkpoint interval will be lost whenever an error is detected, and more intervals may be lost on the occasion of multiple error occurrences.

This paper studies low-cost approaches to mitigate the impacts of soft errors at data caches in mobile video encoding system. Figure 1 shows our system model – *a mobile video encoding system*. A mobile video encoding system consists of several abstraction layers such as application, middleware, OS, and hardware layer. And the mobile video encoding system transmits video data through the wireless network as shown in Figure 1. Due to intensive complexity of processing algo-

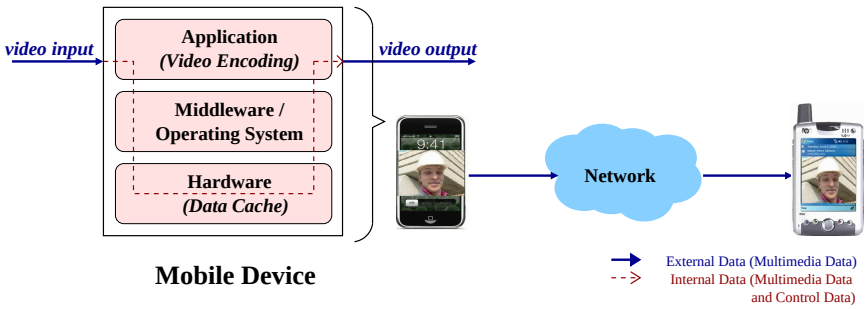


Figure 1: System Model - Mobile Video Encoding System

rithms and large amount of data transmission, it is a challenging task to satisfy multiple constraints such as energy consumption and QoS that mobile video systems demand on battery-operated mobile devices. One of promising approaches to balance these multiple constraints is a cross-layer method. With the global view of the whole system, the cross-layer methods obtain the maximal power reduction with the satisfactory QoS (GRACE) [9, 41], present a proxy-based middleware approach (DYNAMO) by trading off video QoS [7, 8, 21, 22], and recently study online timing-QoS verification at the proxy server (xTune) [14]. To the best of our knowledge, no efforts have studied the cross-layer interactions and potential cooperations among error control schemes to maintain multiple constraints in resource-constrained mobile devices.

In this work, we analyze different error models and error control schemes across abstraction layers, and observe that they have different impacts on reliability and QoS. Soft errors can degrade not only the video quality but also reliability while packet losses cause quality degradation, but not system failures such as system crash, infinite loop, memory violation, etc. Thus, we present a cross-layer, error-aware method in mobile devices so that it protects hardware components such as data caches from soft errors for satisfactory QoS and reliability with minimal costs of power and performance. To mitigate impacts of soft errors on data caches for low costs, we wisely exploit an error-resilient video encoding and a DFR (Drop and Forward Recovery) mechanism with an EDC protection in mobile video encoding systems. Since TMR and ECC incur overheads of power and performance, less expensive EDC is implemented in the previously proposed PPC (Partially Protected Caches) [17] at the hardware layer. A DFR mechanism is selected for the reliability improvement rather than a backward error recovery (BER) once an error is monitored at the middleware layer. Note that a DFR mechanism is named from a simple error concealment scheme in video decodings so that it drops a lossy frame due to transmission errors, and reconstructs it by making use of available data such as adjacent frames [36]. Thus, this DFR mechanism skips the intensive processing algorithms and moves forward to the next frame encoding. This moving-forward is the difference between a DFR and a BER. One potential problem of a DFR mechanism can degrade the video quality since it drops a frame once an error is detected. However, error-resilient techniques in video encodings are wisely exploited to mitigate the impacts of soft errors on the video quality by considering these errors as packet losses. In order to use schemes from other abstraction layers, there are needs for middleware, which allows cross-layer tradeoffs. Middleware translates an error metric at the hardware layer to a different error metric, which is necessary for the use of an error

control scheme at the application layer. For example, SER (Soft Error Rate) at the hardware layer is translated into FLR (Frame Loss Rate), which will be provided to an error-resilient encoding at the application layer. Also, middleware assists a DFR mechanism, and selects a policy based on available information for further improvements.

The contributions and results of our work are:

- We propose a cross-layer, error-aware method so that both performance and energy costs are minimized while obtaining high reliability at the minimal degradation of QoS.
- Our cross-layer method exploiting an error-resilient video encoding and a DFR mechanism with a PPC architecture does not incur overheads in terms of power and performance. Rather, our proposal reduces the access latency of memory subsystem by 61%, the energy consumption of memory subsystem by 52%, the failure rate by about 1000$\times$ at the cost of less than 1 dB of video quality, compared to a traditional video encoding running on a data cache without protection.
- Our cross-layer method extends the applicability of existing error control schemes at the application abstraction layer to mitigate the impact of hardware defects at the hardware abstraction layer.
- To assist our cross-layer methods, we present the middleware that triggers error control schemes with an appropriate error translation, and selects a recovery policy based on available information in a mobile multimedia system.

2 Background

2.1 Hardware Defects and Solutions

Transient hardware defects such as soft errors are emerging problems as technology scales. The primary source of soft errors in digital CMOS (Complementary Metal-Oxide-Semiconductor) circuits are cosmic radiation. Radiation-induced soft errors have been under investigation since late 1970s. Due to incessant technology scaling (voltage scaling and critical dimension scaling), SER has exponentially increased [11, 38]. And in emerging pervasive computing environments, systems will be exposed to drastic increase of radiation intensity [10]. For example, SER in an airplane can be worse than that on the ground by at least a couple of orders of magnitude [20]. Now it has reached a point, where it has become a real threat to system reliability. Solutions to reduce the failures due to soft errors have been proposed at all levels of design hierarchy from hardening devices [3, 29] to microarchitectural solutions [17, 19, 24]. However, these techniques incur the high expense of yield loss, and power and performance overheads.

The error recovery techniques can be classified into Forward Error Recovery (FER) and Backward Error Recovery (BER) [27]. Examples of FER include TMR and ECC [27] where we correct the detected errors to the extent that algorithms can support. However, they are expensive in terms of power and performance [18, 19] since every access, for example, to a data cache memory equipped with an ECC scheme incurs the coding or decoding procedure to insert the redundancy or to correct

as well as detect errors using the redundant information, respectively. On the other hand, recovery with checkpoints [27] are a typical example of BER. This technique should be incorporated with error detection codes (EDC) such as parity codes [27] to at least detect errors in the system. Note that the overheads of EDC in terms of power and performance is much smaller than those of ECC as presented in [19]. However, these BER techniques are not appropriate in real-time embedded systems since it rolls backward at every error detection. And more periods may be lost on the occasion of multiple error occurrences.

Interestingly, multimedia applications have natural features we can exploit to maximize the resource efficiency, especially in battery-operated mobile embedded systems. For example, the inherent robustness of video data enables data cashes to be exposed to soft errors, and can be exploited to present unequal data protection in a PPC (Partially Protected Caches) architecture [17]. Further, exploiting the inherent tolerance of video data itself can be used actively by intentionally injecting errors to maximize the reduction of energy consumption as in [16]. Thus, the error tolerance of video data can be actively exploited to allow a DFR as an affordable mechanism rather than a BER or FER by moving forward to the next correct state when an application or a system meets an error with the minimal costs of protection.

2.2 Error-Resilient Applications

Researchers have studied algorithms in video encodings, e.g., H.263 [13] and MPEG [23], to satisfy multi-dimensional constraints such as QoS, power, and resilience.

One of the most effective methods to achieve the error-resilient video against transmission errors is to introduce the intra-coded frame (I-frame) periodically: since I-frames are decoded independently, they protect the propagation of the transmission errors and even encoding errors in previous frames. However, the transmission of I-frames causes delay and jitter (due to relatively large size) compared to predictively-coded frames (P-frames), and the loss of I-frames is more sensitive on QoS than that of P-frames [5, 15]. To mitigate both the propagation of the transmission errors and the overheads of large I-frames, recently intra-MB (Macroblock) refresh approaches have been proposed [5, 15, 37]. Intra refresh techniques distribute intra-MBs among frames, which not only removes the overheads of I-frames but also improves the error-resilience. While most intra-MB refresh techniques have been focused on alleviating the effects of the transmission errors on the video quality, Kim et al. [15] proposed an energy-efficient and error-resilient video encoding technique named PBPAIR (Probability-Based Power Aware Intra Refresh), and presented tradeoffs between compression efficiency and energy efficiency according to the error resilience for video encoding. However, existing error-resilient video encodings have mostly focused on how to mitigate the impact of network errors on the video quality. They did not address the system failure issues at the mobile devices since network errors clearly do not cause system failures such as a system crash and memory segment violation whereas hardware defects cause failures.

2.3 Cross-Layer Methods

Existing work already demonstrates the effectiveness of cross-layer methods for mobile multimedia as opposed to schemes isolated at a single abstraction layer [7, 8, 9, 14, 21, 22, 41]. Yuan et al.

in [40] proposed an energy-efficient real-time scheduler (GRACE-OS) based on statistical distribution of application cycle demands, and presented a practical voltage scaling algorithm (PDVS) [41] to coordinate adaptation of multimedia applications and CPU speeds for mobile multimedia systems. Mohapatra et al. in [22] presented an integrated power management technique considering hardware-level power optimization and middleware-level adaptation to minimize the energy consumption while maintaining user experience of video quality in mobile video applications. Recently, Kim et al. [14] proposed a unified framework that allows coordinated interactions among sub-layer optimizers through constraint refinement in a compositional cross layer manner to tune the system parameters.

Cross-layer methods in the OSI (Open Systems Interconnection) reference model have been widely investigated as a promising optimization tools to efficiently reduce the energy consumption, especially transmission energy consumption, in wireless multimedia communications [1, 6, 28, 32, 34, 35]. Vuran et al. in [35] presented a cross-layer methodology to analyze error control schemes with respect to transmission power and end-to-end latency, especially impacts of routing, medium access, and physical levels in wireless sensor networks. Schaar et al. in [34] proposed a joint cross-layer approach of application-layer packetization and MAC-layer retransmission strategy, and developed on-the-fly adaptive algorithms to improve the video quality under the bandwidth and delay constraint for wireless multimedia transmission. Bajic in [1] developed cross-layer error control schemes considering joint source rate selection and power management for wireless video multicast.

Our work is novel in two aspects. First, we address a broader notion of reliability than has been explored for error-resilient multimedia applications by specifically focusing on hardware induced defects and their impacts. As illustrated earlier, this issue is a leading concern for embedded architectures of the future. Secondly, we will show how to exploit the cross-layer methodology to activate error control schemes at one abstraction layer to combat errors at a different abstraction layer.

3 A Cross-Layer Approach to Support Reliability and QoS

3.1 System Model and Problem Definition

In the previous section, we argued the need for reliability in mobile applications. The impact of soft errors in memory is particularly relevant for mobile multimedia applications which (a) exhibit higher potential for radiation induced soft errors on outdoor mobile devices, and (b) inherently have a higher potential for memory related errors due to large number of data movements caused by the sheer volume of multimedia information. In addition, mobile multimedia applications such as video streaming and conferencing applications have soft real-time constraints on data delivery. For example, missing deadlines in video streaming applications results in the service delay, and losing packets degrades the video quality – in practice, such degradation (when perceivable) us acceptable to some extent by end-users based on the nature of the application. While we can exploit the soft real-time nature of these applications and its tolerance to slight quality degradation, our ability to do so is already limited in the mobile execution environment that is resource-constrained (limited

buffering and power, error-prone networks).

In this paper, our goal is to exploit the limited error tolerance to enhance the reliability of mobile multimedia applications to hardware-level ”failures” without creating an adverse impact on power/performance profile at the device level or sacrificing on application QoS. We believe that addressing such power/performance/reliability/QOS tradeoffs in the presence of hardware failures requires a cross-layer approach. Firstly, we need to develop an understanding of how errors occur at the various layers and understand existing mechanisms that have been developed to avert errors. This will then enable us to determine when ”errors” become ”failures” and how ”failures” manifest themselves at various system layers. We can then design appropriate schemes at different layers to prevent/bypass specific failures and detect/recover from them. Figure 2 shows our cross-layer model for a mobile video application.

Techniques have been developed to enable QoS in multimedia applications executing in error-prone networks. At the application layer, error-resilient video encoding techniques enable adaptive encoding of information based on knowledge of network conditions[5, 15]. In-network techniques selectively tag data with their level of importance and selectively drop information at different points in the networks when system or network conditions change. Note that these techniques aim to protect the multimedia content that flows through error-prone networks. We refer to this multimedia content as *external data*, i.e., the payload on which the application is executed as shown in Figure 2. In contrast, *internal data* is defined as data, program code, etc. residing inside the mobile device during the process of execution and represents the programs/data that implement the application functionality, e.g., the video codec and associated data/variables. Note that external data may be internal data since it may serve as the input source for the processing and output resides temporarily inside the mobile device before being transmitted outside for the further usage.

The key issue is that while errors in external data (due to packet losses, etc.) only cause quality degradation of the multimedia stream, errors in internal data may cause not only QoS degradation but also system failures. In particular, defects induced at the hardware layer, e.g., data caches or logic components, manifest themselves differently as compared to network errors on external data. For example, errors on program variables may result in memory segmentation violation, which is a system failure. In general, errors on internal data, especially on control data, can result in system crashes, infinite loops, and memory segmentation faults.

Given that hardware is prone to errors that can consequently lead to application failure, error-protection techniques can be designed to protect internal data from hardware failures. We will specifically focus on transient hardware faults (soft errors), i.e., those that do not immediately cause a permanent failure of the system[1]. Traditional protection techniques such as TMR and ECC [27] implemented at the hardware layer to combat such transient errors incur significant overheads in terms of power, performance, and yield cost. For instance, our prior work (PPC or Partially Protected Caches) [17] utilizes knowledge of content and device hardware capabilities to selectively place critical data in more reliable hardware (e.g. a protected cache), but it still incurs overheads of performance and power in the protected cache.

Table 1 presents different error models and error control schemes at the application and hardware abstraction layers in a mobile multimedia system. Exploiting error control schemes across layers is

[1]We also scope out the impact of software bugs introduced by programmers at the application/middleware/OS layers

an interesting challenge since we must consider different types of errors, data, impacts, and error measures as shown in Table 1. By being aware of error specifics and error control schemes, we expect that systems can be designed in a cross-layered manner for obtaining low-cost reliability while maintaining the QoS.

A closer look at Table 1 reveals that while errors occur dynamically and in a transient fashion, techniques to combat these errors may be static or dynamic. For instance, the PPC approach uses compiler-assisted techniques to statically tag data; the operating system uses the tags at runtime to stage the data appropriately into a protected cache. Expensive ECC mechanisms are then employed on the protected data cache to ensure the reliability of information stored in the cache, irrespective of whether the error rate is high or low. Dynamic schemes periodically checkpoint memory state and use knowledge of current error levels, captured via the SER metric, to trigger rollback to the checkpoints. Given the dynamic nature of multimedia data and real-time needs of multimedia applications, this approach as a sole method to deal with soft errors requires very frequent checkpointing and is hence impractical.

3.2 The Cooperative Cross-Layer Approach

We conjecture that a dual pronged approach is needed to effectively address the aforementioned power/performance/reliability/QoS tradeoffs. Firstly, error-monitoring is critical to selectively trigger reliability mechanisms when errors occur. Secondly, the monitored errors are used to tailor intelligent compositions of error-protection schemes across layers in an error-aware manner. There exist inexpensive ways to do an error detection in hardware – output is SER. Specifically, we focus on transient faults (soft errors), i.e., they do not immediately cause a permanent failure of the system. To create error-awareness, we consider the presence of inexpensive error detection mechanisms for soft error detection - these schemes generate as output the soft error rate (SER), which is translated into an error rate for error control schemes described in Section 4.1.2. Hence, our problem is to develop cross-layer methods that, given dynamic soft error rates, are capable of: (i) minimizing the overheads of power and performance, (ii) satisfying the QoS requirement, and (iii) achieving the same level of fault tolerance as traditional error protection techniques. In particular, we investigate techniques to exploit error-resilient video encoding mechanisms (at the application layer) and selective DFR mechanisms (applied at the middleware layer) to reset potentially harmful

Table 1: Error Models and Error Control Schemes at Different Abstraction Layers

	APPLICATION	HARDWARE
Error Model	Packet Losses	Soft Errors
Data Perspective	External Data	Internal Data
Causes	Congestion and Noise	External Radiation
Impacts	Quality Degradation	Quality Degradation and System Failure
Protection	Error-Resilience and Error-Concealment	Forward Error Recovery (ECC) and Backward Error Recovery (Checkpoints)
Error Metric	Packet Loss Rate (%)	Soft Error Rate (FIT)
Time Perspective	Dynamic	Temporary

FIT (Failures In Time): the number of failures in one billion operation hours

data in memory (at the hardware layer).

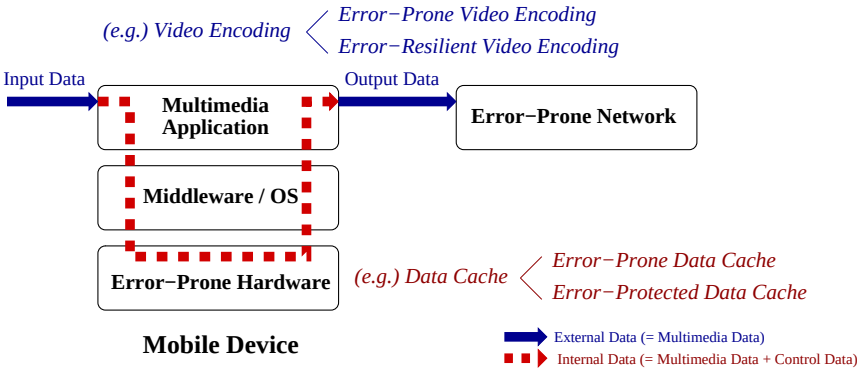


Figure 2: Error-Resilience for External Data in Error-Prone Networks and Error-Protection for Internal Data in Error-Prone Hardware

To illustrate and evaluate our cross-layer approach, we consider a simplified system consisting of a video encoding application and a data cache as shown in Figure 2. A video encoding application can be *error-prone* or *error-resilient*, and similarly a data cache can be *error-prone* or *error-protected* as shown in Figure 2. Any composition of cross-products from them has pros and cons with respect to performance, power, QoS, and reliability. For example, an error-prone video encoding running on an error-prone data cache suffers from high failures due to no protection at data cache against soft errors. While an error-prone video encoding on an error-protected data cache improves the video quality as well as the reliability, it incurs high overheads in terms of power and performance. An error-resilient video encoding on an error-prone data cache may increase the video quality, but fail to increase the reliability. An error-resilient video encoding running on an error-protected data cache is possibly of over protection on the QoS since it incurs high overheads due to expensive protection. Approaches unaware of errors on different data at different abstraction layers may result in inefficiency in terms of power, performance, QoS, and/or reliability.

Cross-Layer, Error-Aware Method (Our Proposal) Our proposal is aware of different data, error control schemes, and impacts across layers. Thus, our cross-layer, error-aware method mitigates hardware defects such as soft errors using error-resilience and a DFR mechanism for maximal reliability with minimal overheads of power and performance at the cost of minimal quality degradation. Given the ability to support error-awareness through less expensive EDC schemes, our strategy is to use the information on SER (soft error rate) to

1. bypass potential failures by triggering error recovery mechanisms which reinitialize the erroneous data cache, and simultaneously

2. reinforce application data using error-resilient encoding mechanisms by translating the SER into the input metric of the encoding algorithm (being considered as the network packet loss rate).

In other words, awareness of micro-level errors (i.e., bit errors) is translated into policies that have macro-level impacts in terms of execution failure, performance, and QoS.

In particular, we explore a Drop and Forward Recovery (DFR) mechanism (shown in Figure 4(c)) that drops a current encoding frame and moves forward to the next frame once an error is detected in a mobile video encoding system. The DFR mechanism works effectively with an EDC scheme to improve power and performance significantly while increasing reliability as well. As discussed, EDC is much less expensive than ECC [19] and overheads due to checkpoints are negligible [39] while EDC can be as immune to soft errors as ECC with respect to reliability. In addition, dropping an erroneous frame potentially improves performance and energy reduction since it skips expensive processing algorithms for encoding the frame.

However, just using DFR-based mechanisms can result in video quality degradation since erroneous frames are actually dropped. To some extent, these errors can be recovered by wisely injecting error-resilience at the application layer. To enhance QoS, we also explore the selective use of Backward Error Recovery (BER) mechanisms that rolls backward and re-encodes the current frame once an error is detected as shown in Figure 4(b).

4 Cooperative Cross-Layer Strategies for Error Resilience

In this section, we present specific CC-PROTECT - a middleware driven approach for cooperative composition of cross-layer strategies to support error resilience.

4.1 CC-PROTECT – A Middleware Driven Strategy for Failure Handling

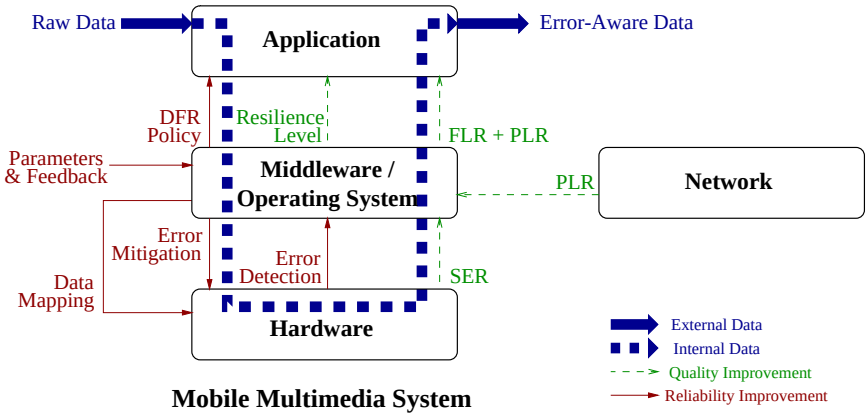


Figure 3: CC-PROTECT – A cross-layer, error-aware method mitigating hardware defects with minimal costs by using error-resilience and a drop and forward recovery in a video encoding

Figure 3 illustrates our CC-PROTECT scheme which exploits the error-resilience of video encoding along with DFR-based error recovery mechanisms to mitigate the impacts of soft errors at the hardware layer. Soft error rates, obtained by error detection techniques at the hardware layer, are communicated to the middleware which then

1. monitors errors, and maintains execution histories and video quality information,

2. translates SER values to corresponding metrics used by other policies (frame loss rate or FLR in our case),

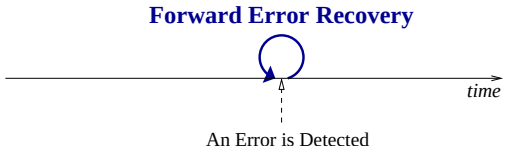
3. initiates DFR/BER policies (discussed later) to avoid and bypass potential hardware failures, and

4. adaptively fortifies multimedia content when hardware errors occur by triggering error-resilient encoding at the application layer.

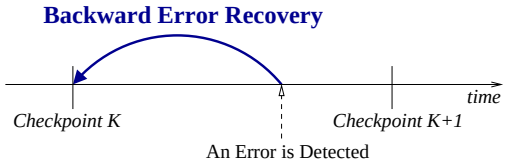
We instantiate two specific strategies for error recovery and error-resilience within CC-PROTECT. The specific error metric we use and evaluate in our study is soft error rate (SER). First, to mitigate the impact of soft errors on the video quality, we exploit a power-aware error-resilient encoding technique, PBPAIR [15]. We present a simple, intuitive and effective translation of SER into frame loss rate (FLR) used in turn by the error-resilient PBPAIR. Next, we exploit our prior work on partially protected caches to design a naive DFR mechanism for PPCs [17]. Using information captured in the middleware, we then extend the naive mechanism to achieve a balance between DFR and BER in Section 4.2.

4.1.1 Drop and Forward Recovery Mechanism for Reliability Improvement

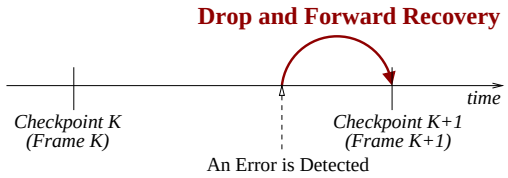
Error recovery techniques can be classified into Forward Error Recovery (FER) and Backward Error Recovery (BER) [27] according to when an error is recovered as shown in Figure 4(a) and Figure 4(b). Figure 4(c) shows the mechanism of DFR. Drop and Forward Recovery (DFR) combines an EDC mechanism with checkpoints to discontinue processing of the current frame and initiate processing of the next checkpointed frame. EDC can only detect an error and is less expensive than ECC in terms of power and performance [19]. Further, a PPC architecture is applied. A PPC consists of two caches at the same level of memory hierarchy such as the unprotected cache and the protected cache for unequal data protection. The protected cache is equipped with EDC instead of ECC to minimize the overheads of power and performance for protecting non-multimedia data. Since multimedia data itself does not cause a system failure [17], multimedia data is exposed to soft errors by being mapped into the unprotected cache in a PPC. Our cross-layer approach maximizes the effectiveness of a PPC architecture by incorporating a DFR mechanism. For DFR, checkpoints are made just before the starting operation where each frame is encoded as exactly same as BER. Only difference is that DFR must save the values for the next frame encoding such as the variables for the next frame, i.e., the information for *Frame K+1* rather than for *Frame K* in Figure 4(c)). Whenever an error is detected onto control data, i.e., non-multimedia data in the protected data cache by EDC at a PPC, it goes to the next frame encoding with the help of the operating system. The DFR mechanism not only helps to increase reliability but also helps to decrease the overheads of power and performance at the slight cost of the quality degradation. Note that a frame drop does not cause a system failure, and further does not cause a significant quality loss mainly due to the inherent error-tolerance of video data [16].



(a) Forward Error Recovery (FER): it detects and even corrects an error



(b) Backward Error Recovery (BER): it rolls back to the last saved state when it detects an error



(c) Drop and Forward Recovery (DFR): it drops a current frame when an error occurs at control data, and moves forward to the next frame in case of a frame-based multimedia processing

Figure 4: Error Recovery Mechanisms

4.1.2 Error-Resilient Video Encoding for QoS Improvement

Error-resilient video encodings have been developed to reduce the impact of transmission errors, e.g., packet losses, on the video quality [5, 15, 37]. The PBPAIR (Probability-Based Power Aware Intra Refresh) technique [15] addresses the tradeoff between energy-efficiency and compression-efficiency according to the resilience level against network errors. Note that PBPAIR is developed to combat network errors, and designed to increase the compression efficiency, i.e., to decrease the encoded file size, at stable network status. We use PBPAIR since it is energy-efficient and its resilience can be adjusted with parameters for various PLRs. PBPAIR takes two parameters such as *PLR* and *Intra_Threshold*. PLR indicates the anticipated error rate in the network and Intra_Threshold can be adjusted by the user expectation of the quality. Since a DFR drops a frame due to soft errors at data caches, this frame loss can be considered the same as a frame loss due to packet losses during transmission. To make use of PBPAIR, our cross-layer approach converts SER for PLR, and selects Intra_Threshold by the original method in PBPAIR. We present a simple conversion. First, the number of soft errors, N_{SE}, during the execution of one frame encoding is calculated as $N_{SE} = S_{cache} \times N_{inst} \times R_{SE}$ where S_{cache} is the size of a cache in KB, N_{inst} is the number of instructions for one frame encoding, and R_{SE} is a SER per instruction per KB. N_{SE} value is then converted to a percent value and used as a FLR (Frame Loss Rate) in our study. For example, S_{cache}

is 32, N_{inst} is 10^8, and R_{SE} is 10^{-10}, then N_{SE} becomes 0.32. So FLR is 32%. Note that FLR becomes 100% if N_{SE} is larger than 1. Now, PBPAIR can generate the compressed video data resilient against the packet losses in networks (PLR) as well as against the soft errors at the hardware layer (FLR) as shown in Figure 3.

4.2 Selective DFR Mechanisms

In a naive DFR approach, any single soft error at the hardware layer causes a frame drop whenever it occurs at the control data (non-multimedia data). However, this approach, named Naive DFR, can significantly degrade the quality in case of consecutive frame drops. To prevent this result, we present intelligent schemes to select a policy between DFR and BER based on the useful information at the mobile embedded system.

4.2.1 Slack-Aware DFR/BER

SA_DFR/BER($S, T_{ACET}, T_{Error}, T_K$**)**
01: $policy = DFR$
02: $T_{elapsed} = T_{Error} - T_K$
03: $T_{threshold} = S \times T_{ACET}$
04: **if** ($T_{elapsed} < T_{threshold}$)
05: $policy = BER$
06: **endIf**
07: **return** *policy*

Figure 5: Slack-Aware DFR/BER Scheme

The main problem with BER is no guarantee to deliver multimedia service in real-time. However, if the remaining time to reach the deadline is enough to re-encode the video frame when an error is detected, we can apply BER rather than DFR for the quality improvement. Since the encoding time is varying from frame to frame and it is hard to measure the remaining time accurately, our scheme presents a knob to select a policy based on the elapsed time with ACET (Average Case Execution Time) as shown in Figure 5. Our knob, S, indicates the portion of ACET, T_{ACET}, that the system can endure. Thus, SA-DFR/BER (Slack-Aware DFR/BER) selects BER if the elapsed time from the starting of the frame K encoding, $T_{elapsed} = T_{error} - T_K$, is smaller than given threshold time, $T_{threshold}$, as shown in Figure 5. Otherwise, SA-DFR/BER selects DFR. For example, if $S = 0.2$ and $T_{ACET} = 100,000$ cycles, $T_{threshold}$ becomes 20,000 cycles. Thus, an error occurring before 20,000 cycles from the starting of the current frame encoding leads to BER. The higher S value increases the probability of BER policy to be selected, and thus improves the video quality while incurring more performance and power overheads due to rolling backward recovery. Indeed, the infinite value of S always causes a BER policy and the zero value of S does a DFR policy.

4.2.2 Frame-Aware DFR/BER

Each frame has a different impact on the video quality. For example, I-frames are considered more important than P-frames with the perspective of the video quality [5, 15]. Thus, if a frame in which

```
FA_DFR/BER(framePara, numFrame, frameType, dropPrev,
diffThresh)
01: policy = DFR
02: switch (framePara)
03:    case IP_FRAME :
04:       if (frameType == I − FRAME)
05:          policy = BER
06:       endIf
07:    break;
08:    case PREV_FRAME_DROPPED :
09:       if (dropPrev == TRUE)
10:          policy = BER
11:       endIf
12:    break;
13:    case DIFFERENCE_FRAMES :
14:       diffFrames = calcDiff(numFrame)
15:       if (diffFrames > diffThresh)
16:          policy = BER
17:       endIf
18:    break;
19: endSwitch
20: return policy
```

Figure 6: Frame-Aware DFR/BER Scheme

a soft error is detected is important in terms of the video quality, FA-DFR/BER (Frame-Aware DFR/BER) rolls back and encodes this frame again (BER) to minimize the quality loss. Otherwise, it drops the current frame and moves forward to the next frame (DFR). Based on available information at the embedded system, the importance of a frame can be decided in several ways. The frame type such as I-frame or P-frame is one example (lines 03-07), and any I-frame will be encoded eventually until no soft error is detected. Another information such as recovery history, e.g., whether the previous frame has been dropped or not due to a soft error, can be used to decide a policy (lines 08-12). If the previous frame was dropped, FA-DFR/BER prevents the current frame from being dropped since the consecutive frame drops may degrade the video quality significantly. Also, the difference between two consecutive frames can be used to estimate the importance of a frame in terms of the video quality. The intuition behind this approach is that the larger difference between two frames indicates the higher impact on the video quality if the current frame is lost. Thus, if the difference between them, $diffFrames$, is larger than given threshold value, $diffThresh$, FA-DFR/BER selects BER (lines 13-18). Otherwise, DFR is selected.

4.2.3 QoS-Aware DFR/BER

The potential problem with a DFR mechanism is the significant degradation of the QoS due to several frame drops. QA-DFR/BER (QoS-Aware DFR/BER) selects a BER policy when the obtained QoS does not meet the QoS requirement. The current quality value, $QoS_{current}$, for frames that have been encoded so far can be calculated at the end of encoding of each frame. QA-DFR/BER selects BER for the erroneous frame if $QoS_{current}$ is worse than $QoS_{threshold}$, given threshold QoS value. Otherwise, the default policy, DFR, is selected. Note that QoS here refers to the encoder, consider-

QA_DFR/BER(prevFrame, threshQoS)
01: $policy = DFR$
02: $QoS_{accumulated} = calcPSNR(prevFrame)$
03: **if** ($QoS_{accumulated} < threshQoS$)
04: $policy = BER$
05: **endIf**
06: **return** $policy$

Figure 7: QoS-Aware DFR/BER Scheme

ing decoder QoS must incorporate knowledge of transmission errors and is beyond the scope of this paper.

5 Experimental Framework

5.1 Simulation Setup

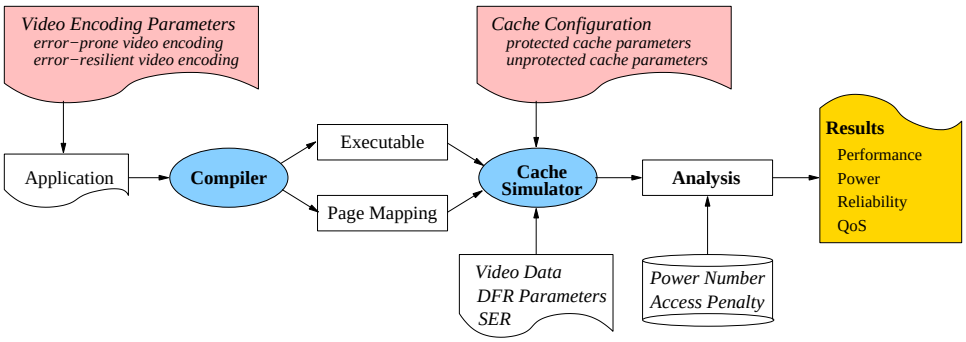


Figure 8: Experimental Setup – Compiler-Simulator-Analyzer Framework

To demonstrate the effectiveness of our cross-layer, error-aware method, a simulation framework has been built as shown in Figure 8. We study a simplified video encoding system consisting of a video encoding at the application layer and a data cache at the hardware layer, and evaluate each system composition in terms of performance, power, reliability, and QoS.

We study an H.263-based error-prone video encoding and error-resilient video encoding. For an error-prone video encoding, we use a GOP (Group-Of-Picture) encoder [5]. For GOP, the first frame is encoded as an I-frame and the other frames are encoded as P-frames, and the quantization scale is set to 10. PBPAIR [15] is used as an error-resilient video encoding that takes two parameters such as Intra_Threshold and PLR. In this study, we consider that the network is error-free, i.e., PLR is set to 0%, to isolate the effects of soft errors from those of network packet losses. Intra_Threshold is selected through the original method of PBPAIR to generate the similar size of the compressed video as GOP in order to ensure a fair comparison – eliminating the overheads of transmission power and delay. Note that we mark all variables for multimedia data (typically, large arrays), and declare them as global variables for data partitioning in a PPC as in [17].

Table 2: System Compositions – Our CC-PROTECT is a middleware-driven, cooperative approach aware of hardware defects

	System Compositions with respect to Error Resilience				
Abstraction Layer	BASE	HW-PROTECT	APP-PROTECT	MULTI-PROTECT	**CC-PROTECT**
Application	GOP *(error-prone encoding)*	GOP *(error-prone encoding)*	PBPAIR *(error-resilient encoding)*	PBPAIR *(error-resilient encoding)*	PBPAIR *(error-resilient encoding)*
Middleware	None	None	○Monitor network errors & Inform PBPAIR of PLR	○Monitor network errors & Inform PBPAIR of PLR	○Monitor network errors & Inform PBPAIR of PLR •**Translate SER to FLR** •**Trigger Selective DFR** **(Drive cache update** **& Inform PBPAIR of FLR)**
Operating system	None	○Map pages to a PPC	None	○Map pages to a PPC	○Map pages to a PPC •**Monitor soft errors**
Hardware	Unprotected Cache *(error-prone cache)*	PPC with ECC *(error-protected cache)*	Unprotected Cache *(error-prone cache)*	PPC with ECC *(error-protected cache)*	PPC with **"EDC"** *(error-protected cache)*

GOP: Group-Of-Picture, PBPAIR: Probability-Based Power Aware Intra Refresh, PLR: Packet Loss Rate, SER: Soft Error Rate, FLR: Frame Loss Rate,
DFR: Drop and Forward Recovery, PPC: Partially Protected Cache, ECC: Error Correction Codes (*expensive in terms of power and performance*, e.g., *a Hamming Code (38,32)*),
EDC: Error Detection Codes (*much less expensive than ECC*, e.g., *a parity code*)

The data cache is modeled using *sim-cache* simulator from the SimpleScalar toolchain [4]. The simulation parameters have been set so as to model an HP iPAQ h5555 [12] like processor-memory system. The *sim-cache* simulator has been modified to support a PPC architecture and to inject soft errors as in [17]. We study a conventional cache (a 32 KB unprotected cache) for an error-prone data cache, and a PPC (a 32 KB unprotected cache + a 2 KB protected cache) for an error-protected data cache. The cache parameters are set with line size of 32 bytes, 4-way set-associativity, and FIFO (First-Input and First-Output) cache replacement policy. The protection technique for a 2 KB protected cache is an ECC or EDC scheme. We use a Hamming Code (38,32) and a parity code for ECC and EDC, respectively. To support the unequal protection for a PPC architecture, compiler as shown in Figure 8 generates not only an executable but also a page mapping table. A page mapping table has a list of the marked global variables (multimedia data), which will be mapped into an unprotected data cache and the other data will be exclusively mapped into a protected data cache in a PPC during simulations. Note that all data will be mapped into an unprotected cache in case of an error-prone data cache.

The modified *sim-cache* simulator runs an executable, i.e., a video encoding, on a given cache configuration with input data such as video streams, SER, and DFR parameters as shown in Figure 8. As test video streams, *AKIYO*, *FOREMAN*, and *COASTGUARD* in QCIF format (176×144 pixels) are used for our simulation study, and each of them represents a video clip of low activity, medium activity, and high activity. To evaluate the cycle accurate results within reasonable amount of simulation time, 300 frames of each video stream are chopped into 75 sequences of four frames (several hours to simulate a video encoding with 300 frames of video on Sun Sparc at 1.5 GHz). For example, 300 frames of *FOREMAN.QCIF* are separated into $FOREMAN_0$*.QCIF*, $FOREMAN_1$*.QCIF*, ..., *and* $FOREMAN_{74}$*.QCIF*. And we ran a simulation at least four times with each sequence, and thus more than 300 runs have been studied ($300\ runs = 4\ times\ of\ run \times 75\ sequences$). DFR parameters are input parameters for selective DFR/BER schemes. For instance, a slack value (S) is given for Slack-Aware DFR/BER in Figure 5.

The simulator models soft errors by randomly injecting single-bit errors and double-bit errors in an unprotected data cache according to SERs. Thus, a single-bit in a data cache is randomly chosen, and a bit value at this single-bit is inverted if a random number generator issues a number less than SER when an instruction is executed in the simulator. Similarly, double-bit errors are injected. Since a protected data cache is resilient against single-bit errors, only double-bit errors occur. To accomplish the experiments in reasonable amount of time, accelerated SERs are used. SER is set to 10^{-11} per KB per instruction for single-bit errors. Note that SER for current technology (2.28×10^{-17} at 90 *nm*)[2] is much less than this accelerated SER by several orders of magnitude, but it increases exponentially as technology scales [2, 11, 20, 38]. However, we maintain the accurate rate (about 10^{-2}) between single-bit SER and double-bit SER, thus 10^{-13} per KB per instruction is used for double-bit errors. The *sim-cache* simulator returns the number of accesses and the number of misses to each cache configuration. We analyzed these statistics with given power and performance numbers, and estimated access time and energy consumption of memory subsystem as shown in Figure 8. QoS is measured in PSNR (Peak Signal to Noise Ratio) with the encoded video

[2]It is projected using the increasing ratio of 1,000 FIT/Mbit at 180 *nm* technology and 100,000 FIT/Mbit at 130 *nm* technology [2, 11, 20].

output and the original video input.

5.1.1 System Compositions

We study a system model consisting of a video encoding and a data cache. A video encoding is GOP (*error-prone*) or PBPAIR (*error-resilient*), and a data cache is an unprotected cache (*error-prone*) or a PPC (*error-protected*). Thus, we compare our cross-layer, error-aware method with the cross-product system compositions from them as shown in Table 2. Main difference between our proposal and other compositions is that our approach is aware of errors and error control schemes across layers. For example, an error-resilient video encoding is aware of hardware defects such as soft errors in our cross-layer approach.

1. **BASE:** This is the default composition, which does not provide any error detection and/or correction. In this composition, we use GOP (Group of Picture) video encoding [5]. For GOP, the first frame is encoded as an I-frame and the other frames are encoded as P-frames, and the quantization scale is set to 10. The middleware and operating system are unaware of soft errors, and hardware has just a unified unprotected cache. Base composition does not incur overheads for protection in terms of power and performance, but suffers from high failure rates and low multimedia quality due to no protection on internal data from hardware defects.

2. **HW-PROTECT:** In this composition, all error detection and correction are provided in hardware. This is implemented through the use of Error Correction Code (ECC) in Partially Protected Cache architecture [17]. As compared to protecting the whole cache, PPCs provide efficient reliability by just protecting the non-multimedia data against soft errors. This composition presents the low failure rate and high QoS, as it protects at hardware level. However, it incurs overheads in terms of power and performance.

3. **APP-PROTECT:** In this composition, all error detection and correction are provided in the application. For this, we use error-resilient video encoding PBPAIR [15]. We set the PLR parameter in PBPAIR to 0% to isolate the effects of soft errors from those of network packet losses. Intra_Threshold is selected through the original method of PBPAIR to generate the similar size of the compressed video as GOP to ensure a fair comparison with respect to the transmission cost.

4. **MULTI-PROTECT:** In this composition, error correction is provided at all levels. We use error-resilient video encoding (PBPAIR) and a protected cache (a PPC with an ECC scheme). It implements both error-resilience at the application layer and an ECC scheme at the hardware layer.

5. **CC-PROTECT:** This is our proposed composition, in which we use error-resilient video encoding, PBPAIR, and PPC with an EDC scheme, and supports middleware-driven mechanisms aware of soft errors such as translating SER for PBPAIR and triggering a hybrid scheme of DFR and BER.

5.1.2 Selective Mechanisms

In order to evaluate our selective schemes such as Slack-Aware (SA) DFR/BER, Frame-Aware (FA) DFR/BER, and QoS-Aware (QA) DFR/BER, we consider Naive DFR, Naive BER, No DFR/BER, and Random DFR/BER. Naive DFR always triggers a DFR mechanism when an error is detected. Similarly, Naive BER always triggers a BER mechanism. No DFR/BER does not trigger any recovery mechanism. Random DFR/BER selects DFR or BER based on randomly generated number, e.g., if the generated number is less than 50 out of 100, it triggers BER. For FA-DFR/BER, SA-DFR/BER, and QA-DFR/BER, we profiled preliminary experiments, and selected parameters for each selective scheme in our simulations.

5.2 Strategy and Evaluation Metrics

We have three main strategies for our experimental study. First, our experiments evaluate different layering approaches as shown in Table 2 in terms of power, performance, reliability, and QoS. Simulation results demonstrate that our cross-layer, error-aware method is very effective to accomplish low-cost reliability at the slight degradation of QoS. Secondly, we study the impacts of heterogeneous video contents as well. Experiments show that our approach maintains the low-cost reliability with minimal quality degradation for various video streams. Finally, we study the impacts of selective DFR/BER polices. Experimental results show the effectiveness of our selective schemes to improve the video quality while still maintaining low-cost reliability of the system.

Therefore, these experimental outcomes open opportunities that guide system designers to effectively explore interesting tradeoff spaces under the multiple constraints for mobile multimedia systems.

5.2.1 Measuring Memory Subsystem Performance

For performance comparison of each composition, we estimate the access latency to the memory subsystem using the statistics generated by the cache simulator as shown in Figure 8. Note that our performance model using a functional simulator *sim-cache* has been compared to runtime using a cycle accurate processor simulator *sim-outorder*, and the relative ratio between them maintains constant, but *sim-cache* is much faster. The access latency of memory subsystem L is estimated as $L = (A_{cache} \times L_{access}) + (M_{cache} \times L_{miss}) + (N_{policy} \times L_{policy})$ where A_{cache} is the number of accesses to a cache, L_{access} is the cache access time, M_{cache} is the number of misses to a cache, L_{miss} is the cache miss penalty, i.e., the access penalty to a bus and a memory, N_{policy} is the number of triggered policies such as DFR and BER, and L_{policy} is the latency penalty for a policy. Table 3 summarizes access and miss penalties of a data cache in cycles with or without a protection. The overhead of delay for ECC is estimated and synthesized using the CACTI [30] and the Synopsys Design Compiler [33] as in [17], and the overhead of delay for EDC is calculated using the ratio between delays of ECC and EDC from [17, 19]. For instance, the cache miss penalty is 25 cycles (including bus access and main memory access), and the cache access penalty for a 2 KB of protected cache with a Hamming code (38,32) is 2 cycles while that with a parity code is 1.5 cycles. Also, the delay overheads for DFR and BER are estimated through the simulations so that the overheads for context

switch and checkpoints are added at the analysis stage in our simulation study as shown in Figure 8.

5.2.2 Measuring Energy Consumption of Memory Subsystem

For power comparison, we estimate the energy consumption of the memory subsystem. To estimate the energy consumption of the memory subsystem, we use the power models presented in [31]. The overheads of power for a Hamming code (38,32) and a parity code are synthesized and estimated similar to those of delay. The power consumption penalty for a recovery policy such as DFR and BER is estimated through the simulations. With the power models in Table 3, the energy consumption of the memory subsystem E as $E = (A_{cache} \times P_{access}) + (M_{cache} \times P_{miss}) + (N_{policy} \times P_{policy})$ where P_{access} is the power consumption per cache access, P_{miss} is the power penalty per cache miss, and P_{policy} is the power penalty for a recovery policy.

5.2.3 Measuring Failure Rate

For reliability evaluation, a failure rate is used. Each execution is defined as a *Success* if it ends within twice of a normal execution time and returns the correct output opened by a decoder. Otherwise, it is a *Failure* such as a system crash, infinite loop, or segmentation fault. Note that the degradation of video data is not considered as a failure in our study. The failure rate has been obtained through at least hundreds of executions for each composition by counting the number of failures out of a total number of executions based on the following binomial distribution analysis.

Assuming that each execution is an independent *Success* /*Failure* event, X is the number of *Success* in executions, and we perform n runs, then if the probability of a failure is p, then X is a binomially distributed random variable, which follows the binomial distribution with parameters n and p ($X \approx B(n,p)$). Therefore, the mean of X is $\mu = np$, and the variance of X will be $\sigma^2 = np(1-p)$. The error $E = |\overline{X} - \mu|$ is less than or equal to $z_{\alpha/2}\sigma/\sqrt{n}$ with confidence $100(1-\alpha)\%$, where α is a confidence level. Therefore, the sample size to be able to state with $100(1-\alpha)\%$ confidence that the error $|\overline{X} - \mu|$ will not exceed a specified amount E is $N = (\frac{z_{\alpha/2}\sigma/\sqrt{n}}{E})^2$. Thus, for 95% confidence, $z_{\alpha/2} = 2$, and confidence interval .01%, the sample size is $N = 40,000p(1-p)$. To estimate the probability of a failure, suppose the probability of a failure is 1%, then $N = 40,000 \times 0.01 \times 0.99 = 396$.

Table 3: Delay and Power Numbers for Caches from [17,19,30,33]

Cache (size)	Protection	L_{access} (cycles)	P_{access} (nJoules)	L_{miss} (cycles)	P_{miss} (nJoules)
Unprotected (32 KB)	None	2	1.06	25	41.96
Unprotected (2 KB)	None	1	0.80	25	41.96
Protected (2 KB)	EDC	1.5	0.91	25	42.06
Protected (2 KB)	ECC	2	1.19	25	42.16

5.2.4 Estimating the Video Quality

For QoS comparison in video encodings, the video quality is typically measured in PSNR (Peak Signal to Noise Ratio). PSNR is defined in dB as $PSNR = 10LOG_{10}(\frac{MAX^2}{MSE})$, where *MAX* is the maximum pixel value and *MSE* is the Mean Squared Error, which is the mean of the square of differences between the pixel values of the erroneous video output (due to soft errors and frame drops), and the correctly reconstructed output (without errors). On the occasion of a frame drop due to soft errors, we considered a simple error-concealment scheme at the decoding side, i.e., the decoder copies the previous frame for the dropped current frame. For example, the video sequence consists of $frame_1$, $frame_2$, $frame_3$, and $frame_4$. If a soft error occurs at the control data (i.e., on the protected cache in a PPC or Partially Protected Cache) when $frame_2$ is being encoded, CC-PROTECT drops $frame_2$ and an error-resilient video encoding takes care of it in terms of video quality. Now, when a decoder will receive just $frame_1$, $frame_3$, and $frame_4$ ($frame_2$ is dropped). Then the decoder decodes $frame_1$, $frame_1$, $frame_3$, and $frame_4$. Simply, the PSNR value is calculated with these frames.

6 Experimental Results

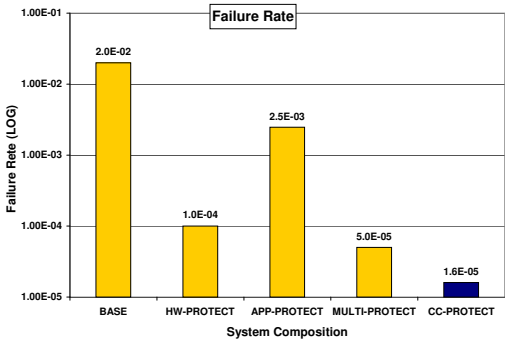
We present two sets of experiments. First, we demonstrate the effectiveness of our cross-layer, error-aware methods in low-cost reliability at the slight cost of QoS (Section 6.1). Second, we show the effectiveness of intelligent DFR/BER selection schemes to improve the video quality (Section 6.2).

6.1 Effectiveness of CC-PROTECT

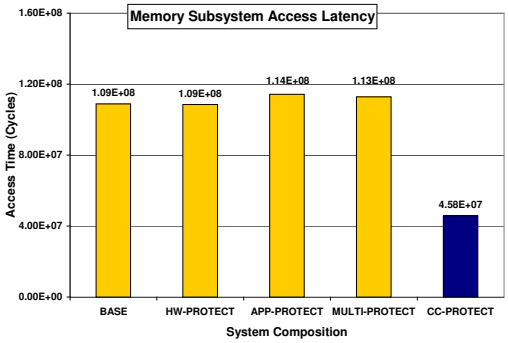
Figure 9 clearly shows that our cross-layer, error-aware approach increases the reliability with the minimal costs of performance and energy consumption at the minimal degradation of video quality.

Figure 9(a) clearly demonstrates that our CC-PROTECT (PBPAIR, a DFR mechanism, and a PPC with an EDC protection) improves the failure rate by more than 1,000 times than that of BASE (GOP and an unprotected data cache). This reliability improvement mainly because of the error detection and a DFR mechanism in a cross-layered manner. While HW-PROTECT and MULTI-PROTECT have lower failure rates than that of BASE, they have higher failure rate than that of CC-PROTECT. This is because CC-PROTECT has less time to be exposed to soft errors due to a frame drop and the performance efficiency of PBPAIR than GOP. It is important that APP-PROTECT (composed of PBPAIR and an unprotected data cache) shows the close failure rate to that of BASE since a failure results from errors on control data, which are not protected in APP-PROTECT. Thus, our CC-PROTECT scheme can achieve the best reliability among all compositions.

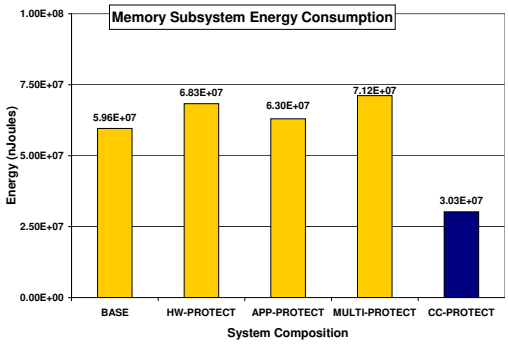
Figure 9(b) shows that our CC-PROTECT is the best in terms of performance. It reduces the memory subsystem access delay by 58%, compared to that of BASE. It is very effective since our CC-PROTECT reduces the failure rate by 1,000 times and it reduces the access latency of memory subsystem compared to BASE. All the other compositions incur the performance overhead but CC-PROTECT improves the performance. This performance improvement is because of skipping intensive compression algorithms from a DFR mechanism, and because of the performance efficiency of PBPAIR algorithms. However, the performance efficiency of PBPAIR is not well exploited in



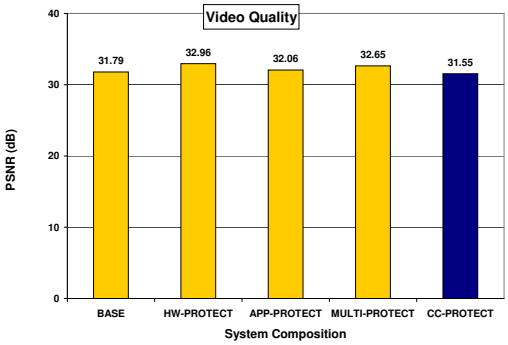
(a) Reliability: Failure Rate



(b) Performance: Access Time to Memory Subsystem



(c) Power: Energy Consumption of Memory Subsystem



(d) QoS: Video Quality

System Composition	BASE	HW-PROTECT	APP-PROTECT	MULTI-PROTECT	CC-PROTECT
Application	GOP	GOP	PBPAIR	PBPAIR	PBPAIR
Recovery Mechanism	NO	FER	NO	FER	DFR
Hardware	Unprotected Cache	PPC with ECC	Unprotected Cache	PPC with ECC	PPC with EDC

Figure 9: CC-PROTECT achieves the low-cost reliability at the minimal QoS degradation

case of APP-PROTECT as it shows 5% overhead compared to BASE because PBPAIR increases the compression efficiency rather than the performance efficiency at low PLR such as 0% PLR. With the same reason, MULTI-PROTECT incurs about 4% overhead compared to BASE. Indeed, HW-PROTECT does not incur the performance overhead because a PPC achieves high performance by protecting only non-multimedia data [17].

With the perspective of energy consumption of memory subsystem, our CC-PROTECT saves the energy consumption by 49%, 56%, 52%, and 57% as compared to BASE, HW-PROTECT, APP-PROTECT, and MULTI-PROTECT, respectively, as shown in Figure 9(c). CC-PROTECT approach reduces the energy consumption of memory subsystem because of (i) less expensive EDC technique than ECC, (ii) skipping expensive compression algorithms thanks to a DFR mechanism, and (iii) energy efficiency of PBPAIR by introducing more intra-MBs (Macroblock) than expensive inter-MBs. Note that all other compositions incur overheads of performance as well as power compared to BASE except for CC-PROTECT. Thus, our CC-PROTECT scheme can even reduce the power

and access time of memory subsystem while obtaining the high reliability. And this is very effective, especially for resource-constrained mobile embedded systems.

Our CC-PROTECT achieves video quality close to those of other compositions as shown in Figure 9(d). While an EDC scheme protects the non-multimedia data in our CC-PROTECT, a frame drop due to a DFR mechanism degrades the video quality. Note that PBPAIR algorithms can improve this video quality by increasing the resilience level at the cost of the compressed video size (causing the transmission costs of power and delay). However, CC-PROTECT saves at least 49% of power and performance for the minimal failure rate at the minimal cost of QoS by up to 1.41 dB (less than 5% quality degradation) compared to all the compositions. Note that these results come from only one soft error at the protected cache in a PPC, and the video quality may degrade significantly due to multiple frame drops resulting from multiple occurrences of soft errors. We will present the experimental results in those cases in Section 6.2.

Table 4: CC-PROTECT is very effective in terms of performance, power, and reliability at the minimal QoS degradation for different video streams (normalized result of each composition to that of BASE)

Video Stream		System Composition	Access Time	Energy Consumption	Failure Rate	Video Quality
AKIYO *(low activity)*		BASE	1	1	1	1
		HW-PROTECT	0.99	1.14	0.4 E-2	1.02
		APP-PROTECT	0.87	0.89	13.2 E-2	1.01
		MULTI-PROTECT	0.89	1.03	0.2 E-2	1.02
		CC-PROTECT	**0.27**	**0.34**	**0.1 E-2**	**1.02**
FOREMAN *(medium activity)*		BASE	1	1	1	1
		HW-PROTECT	1	1.15	0.5 E-2	1.04
		APP-PROTECT	1.05	1.06	12.3 E-2	1.01
		MULTI-PROTECT	1.04	1.19	0.3 E-2	1.03
		CC-PROTECT	**0.42**	**0.51**	**0.1 E-2**	**0.99**
COASTGUARD *(high activity)*		BASE	1	1	1	1
		HW-PROTECT	0.99	1.14	0.4 E-2	1.03
		APP-PROTECT	1.09	1.1	13.0 E-2	0.99
		MULTI-PROTECT	1.06	1.23	0.3 E-2	1.02
		CC-PROTECT	**0.49**	**0.58**	**0.1 E-2**	**0.93**

Table 4 summarizes the normalized results of each composition to those of BASE in terms of performance, power, reliability, and QoS for different video streams. This table clearly shows that CC-PROTECT has the least costs of power and performance for the minimal failure rate with the minimal QoS degradation for heterogeneous video streams. The interesting observation that we can make from this table is that we can even improve the video quality while still saving the performance and power costs (73% and 66%, respectively) compared to BASE for a video stream *AKIYO*. This quality improvement (about 2%) is because: (i) a frame drop may not affect the video quality for a video stream with low-activity such as *AKIYO*, and (ii) less amount of execution time of PBPAIR results in less exposure of a data cache to soft errors. Indeed, the QoS impact of one frame drop for *AKIYO* is about 0.08% on average. On the other hand, for high activity of video stream such as *COASTGUARD* our CC-PROTECT approach degrades the video quality by about 6% in PSNR. But still CC-PROTECT demonstrates the best access time and energy consumption for the

minimal failure rate. Note that all these results are evaluated under the condition of no errors in the network. We observed the similar results under the various network status. For example, at 10% PLR, our CC-PROTECT approach reduces access time of memory subsystem by 58%, while APP-PROTECT saves it by 32% (more error rate triggers more intra-MBs, causing high performance in PBPAIR algorithms), as compared to BASE.

Table 5: Experimental results demonstrate the effectiveness of CC-PROTECT compared to other compositions without cooperation in a cross-layered manner (Normalized result to that of BASE composition, *FOREMAN*).

System Composition	Application	Policy	Hardware	Access Time	Energy Consumption	Failure Rate	Video Quality
Base	GOP	NO	Unprotected Cache	1	1	1	1
HW-PROTECT	GOP	FER	PPC with ECC	1.00	1.15	0.5E-2	1.04
HW-PROTECT-2	GOP	FER	Protected Cache with ECC	1.45	1.34	1.1E-2	1.03
HW-PROTECT-3	GOP	BER	Protected Cache with EDC	32.59	29.71	40.2E-2	1.05
APP-PROTECT	PBPAIR	NO	Unprotected Cache	1.05	1.06	12.3E-2	1.01
MULTI-PROTECT	PBPAIR	FER	PPC with ECC	1.04	1.19	0.2E-2	1.03
MULTI-PROTECT-2	PBPAIR	FER	Protected Cache with ECC	1.41	1.32	4.0E-2	1.05
CC-PROTECT	PBPAIR	DFR	PPC with EDC	0.42	0.51	0.1E-2	0.99
CC-PROTECT-2	GOP	DFR	PPC with EDC	0.63	0.74	0.7E-2	0.96

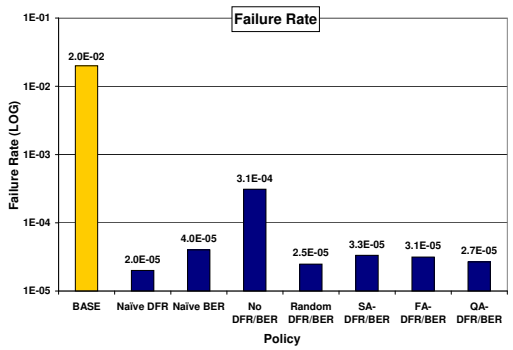
Also, we have run simulations for different compositions, and results demonstrated the effectiveness of our cross-layer protection approach, CC-PROTECT. HW-PROTECT compositions with traditional error-protected caches incur high performance overheads as summarized in Table 4. For instance, HW-PROTECT-2 composition (GOP and a 32 KB of protected cache with an ECC – forward error recovery) incurs 45% performance and 34% energy overheads as compared to BASE as shown in Table 5. This is because all data (multimedia data as well as control data) are protected from soft errors with an expensive ECC scheme. Further, HW-PROTECT-3 (GOP + BER + EDC-based Cache) incurs 31.59 times more access time than BASE due to about 56 errors every execution, which indicates approximately 56 times BER. The similar number of soft errors occur at a PPC cache while most of them occur at the larger unprotected cache and only a couple of errors are detected at the smaller protected cache. Since most errors in multimedia data itself do not cause failures [17], we can save the performance overhead by applying this unequal protection to a PPC architecture. Since the quality degradation results from soft errors at data caches, HW-PROTECT-3 and MULTI-PROTECT-2 (PBPAIR + FER + a 32 KB of protected cache with ECC) compositions present the best video quality by protecting all data from soft errors. Table 4 shows that CC-PROTECT-2 (GOP + DFR + a PPC with an EDC protection) degrades the video quality by 6% (about 1.89 dB) compared to that of BASE mainly because of dropping a frame when it meets a soft error at the protected data cache in a PPC. This quality degradation is the cost of a DFR mechanism. However, our CC-PROTECT with PBPAIR improves the video quality close to that of BASE mainly because of the error-resilience of PBPAIR. Most compositions present a better video quality than our CC-PROTECT while the quality degradation is up to 2.1 dB (less than 6% quality degradation than the best QoS). Note that our CC-PROTECT reduces the access time by 70%, energy consumption by 62% with higher reliability at the cost of less than 5% quality degradation, as compared to HW-PROTECT-2 in Table 4.

In summary, our cross-layer, error-aware methods exploit a DFR mechanism with an inexpensive EDC protection to decrease the failure rate by about 1,000×, and an error-resilient video encoding technique to minimize the quality degradation by 2% while significantly saving the access time by 61% and energy consumption by 52% on average over multiple video streams, as compared to BASE. Also, our cooperative, cross-layer approach (CC-PROTECT) achieves a better reliability than a previously proposed PPC architecture with an ECC protection at the cost of 3% QoS degradation while reducing the access time by 60% and the energy consumption by 58% on average. These results are extremely effective.

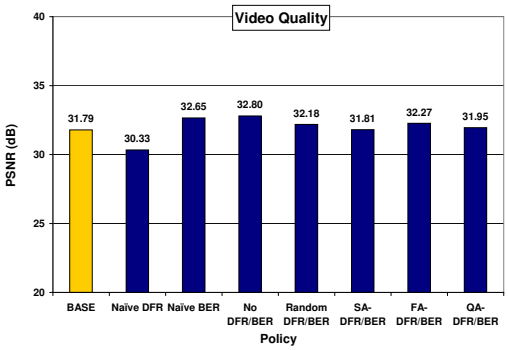
6.2 Effectiveness of Intelligent Selective Schemes

Our CC-PROTECT outperforms all possible compositions in terms of power, performance, and reliability while it degrades the video quality mainly due to frame drops when soft errors occur. Figure 10 demonstrates that all intelligent selective schemes improve the video quality without incurring performance and energy costs significantly (still mostly lower than costs of BASE).

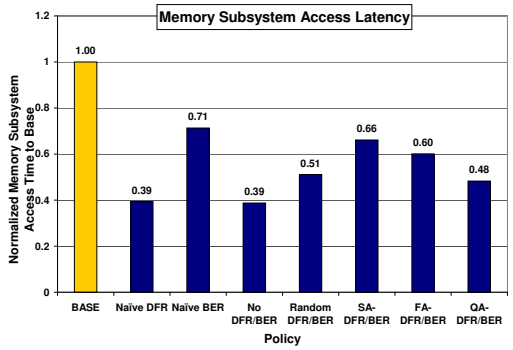
In Figure 10, X-axis represents selective mechanisms compared to BASE. Note that they are all running PBPAIR on a PPC architecture with an EDC scheme except for BASE (running GOP on an unprotected cache) and No DFR/BER (running PBPAIR on a PPC without any protection, i.e., a PPC consisting of a 32 KB unprotected cache and a 2 KB unprotected cache) for comparison. And we parameterize a policy selection based on available information in a mobile embedded system at the moment when a soft error is detected. Naive DFR always selects a DFR mechanism while Naive BER selects a BER once a soft error occurs. Naive DFR scheme shows the worst video quality as shown in Figure 10(b) at the minimal costs in terms of power and performance as shown in Figure 10(c) and Figure 10(d). Note that Naive DFR in Figure 10 results from multiple soft errors (1.7 errors on average) on the protected data cache in a PPC, which degrades the video quality worse than that of CC-PROTECT in Figure 9(d) (results from single soft error on the protected cache in a PPC). On the other hand, Naive BER scheme presents the better video quality than that of Naive DFR while incurring the most expensive power and performance costs. In terms of reliability, Naive BER shows higher failure rate than that of Naive DFR as shown in Figure 10(a). This is mainly because Naive BER increases the execution time, causing the more time for a PPC to be exposed to soft errors. Clearly, No DFR/BER does not have a mechanism to protect a system from soft errors, causing very high failure rate as shown in Figure 10(a). Note that Figure 10(b) shows higher video quality of No DFR/BER than others. This is because we measured the video quality in PSNR when simulations are *Successes* where No DFR/BER does not skip any frame. Random DFR/BER provides the good video quality with inexpensive costs of power and performance. For SA-DFR/BER, the results have been profiled with the knob S, the portion of ACET, from 0% to 100% in 10% increments, and SA-DFR/BER with S = 60% is compared in Figure 10 since it is the least value of the knob to recover the video quality better than that of BASE according to profiled results. However, it is an expensive approach since it incurs high overheads in terms of power and performance while it presents a better video quality than that of Naive DFR. For FA-DFR/BER scheme, our preliminary experiments show that the difference in PSNR between consecutive frames make the 2^{nd}, 3^{rd}, and 4^{th} frame in the descending order of the importance on average. Thus, FA-DFR/BER with 2^{nd} frame is studied to improve the video quality most and indicates that we select



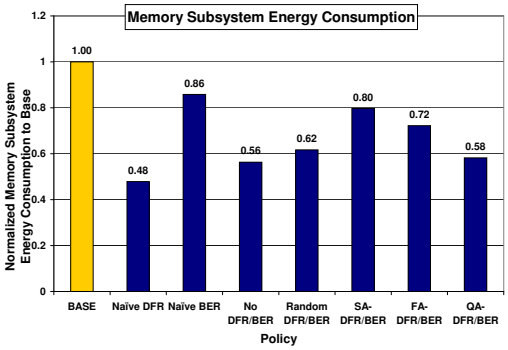
(a) Reliability: Failure Rate



(b) QoS: Video Quality



(c) Performance: Access Time to Memory Subsystem



(d) Power: Energy Consumption of Memory Subsystem

Selective Scheme	BASE	Naive DFR	Naive BER	No DFR/BER
Video Encoding	GOP	PBPAIR		
Recovery Policy (threshold value)	None	Always DFR	Always BER	None
Data Cache	Unprotected Cache	PPC with EDC		PPC (No Protection)

Selective Scheme	Random DFR/BER	SA-DFR/BER	FA-DFR/BER	QA-DFR/BER
Video Encoding	PBPAIR			
Recovery Policy (threshold value)	Randomly DFR or BER (50%)	DFR or BER (Slack>60%)	DFR or BER (Frame$\neq 2^{nd}$)	DFR or BER (QoS>31.79 dB)
Data Cache	PPC with EDC			

Figure 10: Intelligent selective schemes maintain the video quality and reliability with minimal overheads of power and performance

BER rather than DFR whenever a soft error occurs in encoding the 2^{nd} frame. In these particular experiments, FA-DFR/BER scheme is more effective than SA-DFR/BER scheme since it has lower costs with better QoS (failure rates are close). For QA-DFR/BER, 31.79 dB is considered as the QoS threshold value since it is the average video quality in case of BASE. QA-DFR/BER provides lower video quality while incurring less costs than FA-DFR/BER scheme. Thus, each selective scheme has pros and cons in terms of performance, power, reliability, and QoS.

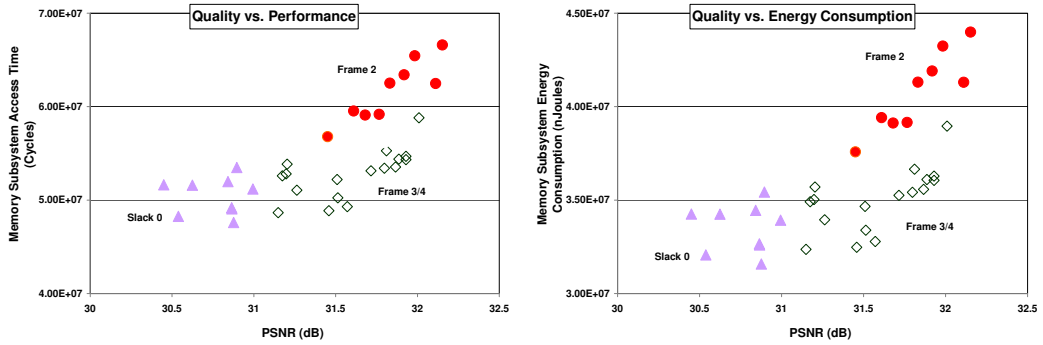
In summary, selective DFR/BER mechanisms allow a system to maintain the video quality and reliability with minimal costs of power and performance.

7 Discussion

One of main contributions in this paper is to exploit existing error control schemes such as a DFR and PBPAIR. By cooperating them with an inexpensive error detection technique, e.g., EDC, mobile devices obtain high reliability, high performance, and high energy saving at the slight cost of QoS degradation. This extension by existing techniques introduces new opportunities that system designers can consider for resource-constrained mobile devices. For example, instead of implementing an ECC scheme to protect data caches from soft errors, monitoring errors with an EDC scheme and triggering available schemes can be very effective for battery-operated mobile video encoding systems as we demonstrated. Also, by trading off costs of power and performance for QoS, selective DFR/BER schemes can be used adaptively to balance these multiple constraints. Furthermore, combined approaches with several selective DFR/BER schemes can expand the interesting tradeoff spaces for system designers.

In order to explore interesting design spaces, we ran intensive simulations of combined selective mechanisms with SA-DFR/BER, FA-DFR/BER, and QA-DFR/BER for 4 frames of a video sequence. A slack knob (S) for SA-DFR/BER ranges 0%, 20%, 60%, and 100% since 20% and 60% for a knob profiled interesting results. Each frame such as 2^{nd}, 3^{rd}, and 4^{th} out of 4 frames is considered as a threshold frame for FA-DFR/BER, and the 1^{st} frame is excluded since 1^{st} frame loss degrades the QoS significantly. The quality threshold values include 31, 32, and 33 dB for QA-DFR/BER. Thus, 36 combinations (= $4 \times 3 \times 3$) were explored by our cross-layer methods, which have never been explored with compositions without considering error-awareness across layers. The interesting observation we make from these simulations is that we can effectively explore the tradeoff spaces according to the purpose of designs. In fact, combined approaches with this small set of threshold values expand the design space to improve the performance by up to 29% further at the cost of less than 2 dB QoS degradation.

Figure 11(a) and Figure 11(b) present these 36 combinations in video quality vs. memory subsystem access time and in video quality vs. energy consumption of memory subsystem, respectively. The graphs show similar trends since high memory access time incurs high energy consumption of memory subsystem. Each dot indicates video quality in PSNR with access time (or energy consumption) for a combined selective scheme. For example, a dot with highest performance overhead and highest quality (right-top circle) in Figure 11(a) presents a combined scheme such that BER is selected if a soft error occurs within 100% portion of ACET at 2^{nd} frame for 33 dB quality requirement, i.e., 100% slack knob for SA-DFR/BER, 2^{nd} frame for FA-DFR/BER, and 33 dB for



(a) Video Quality in PSNR vs. Access Time to Memory Subsystem

(b) Video Quality in PSNR vs. Energy Consumption of Memory Subsystem

Figure 11: Design Space Exploration

QA-DFR/BER. This combination induces the highest probability to select a BER rather than a DFR, and generates the best video quality while it incurs the highest overheads in terms of performance and energy consumption. On the other hand, the combinations with a slack knob = 0% shows worse video quality and better performance (or energy consumption) since SA-DFR/BER with 0% knob of slack drives a mechanism to select a BER once a soft error occurs. Thus, the interesting observation we can make from these graphs is that we can explore interesting tradeoff spaces by choosing one policy from BER and DFR with selective schemes according to the purpose of designs. 9 triangle (△) dots in Figure 11(a) and Figure 11(b) indicate slack 0% combinations with 3 different frames and 3 different quality thresholds, and they are all of lower quality and improved performance (or reduced energy consumption). In this study, FA-DFR/BER affects the design space most effectively such that it trades off the performance and energy consumption for the video quality. Figure 11(a) and Figure 11(b) clearly show that combinations with 2^{nd} frame for FA-DFR/BER generates higher overheads in terms of power and performance with better quality as represented by circles (○) in these graphs. In contrast, the other combinations (◇ dots) with 3^{rd} and 4^{th} frame present lower quality but better performance (or energy reduction) than those with 2^{nd} frame as shown in Figure 11.

Also, we studied Pareto-optimal cases. Out of 36 combinations, 11 are Pareto-optimal, which are interesting design points for system designers. A combination is Pareto-optimal if it is no worse than any other combinations in all dimensions, e.g., performance, energy consumption, and video quality. This Pareto-optimal observation enables system designers to select cross-layer methods combined with selective DFR/BER mechanisms for their purposes.

In summary, our cross-layer methods aware of error control schemes and errors can guide system designers to explore the interesting tradeoff spaces in terms of power, performance, reliability, and QoS for resource-constrained mobile devices, which were not discovered by previous approaches.

Conclusion Remarks

Reliability is a paramount concern in mobile embedded systems where the resources such as power and performance are limited. In order to resolve the complexity of trade-offs among multi-dimensional properties, a cross-layer approach from the hardware layer to the application layer should be taken into account since traditional techniques are unable to address the impacts of an

approach on other properties at other layers, and are unable to drive the whole system's reliability in a power and performance efficient way.

We present a cross-layer, error-aware method exploiting existing error control schemes to mitigate the effects of soft errors in mobile video encoding system: (i) an occurrence of a soft error at data caches triggers a DFR mechanism for reliability, and (ii) SER is translated to exploit the error-resilience of PBPAIR for QoS improvement. We demonstrate that our approach with selective schemes between DFR and BER is very effective to achieve low-cost reliability while maintaining the QoS for mobile video applications. Thus, our cross-layer, error-aware methods not only extend the applicability of existing error control schemes but also open opportunities that system designers can expand their design spaces for multiple system constraints such as performance, power, reliability, and QoS. Note that our cross-layer, error-aware methods can be easily applied for any hardware components, e.g., logic circuits, incorporated with inexpensive error detection schemes rather than error correction schemes.

Our future work includes the extended cross-layer approach considering end-to-end devices in distributed real-time systems, and considering various error control schemes with different error models across system abstraction layers. Also, intelligent design space algorithms will be investigated to efficiently guide system designers for exploiting our cross-layer schemes.

References

[1] Ivan V. Bajic. Efficient cross-layer error control for wireless video multicast. 53(1):276–285, Mar 2007.

[2] Robert Baumann. Soft errors in advanced computer systems. *IEEE Design and Test of Computers*, pages 258–266, 2005.

[3] Mark P. Baze, Steven P. Buchner, and Dale McMorrow. A digital CMOS design technique for SEU hardening. *IEEE Trans. on Nuclear Science*, 47(6):2603–2608, Dec 2000.

[4] Doug Burger and Todd M. Austin. The SimpleScalar Tool Set, version 2.0. *SIGARCH Computer Architecture News*, 25(3):13–25, 1997.

[5] Liang Cheng and Magda El Zarki. PGOP: An error resilient techniques for low bit rate and low latency video communications. In *Picture Coding Symposium (PCS)*, Dec 2004.

[6] Jeong-Yong Choi and Jitae Shin. Cross-layer error-control with low-overhead ARQ for H.264 video transmission over wireless LANs. *Computer Communications*, 30(7):1476–1486, 2007.

[7] DYNAMO. *Power Aware Middleware for Distributed Mobile Computing*. University of California at Irvine, http://dynamo.ics.uci.edu/.

[8] FORGE Project. *A Framework for Optimization of Distributed Embedded Systems Software*. University of California at Irvine, http://www.ics.uci.edu/ forge/.

[9] GRACE Project. *Global Resource Adaptation through CoopEration*. University of Illinois at Urbana-Champaign, http://rsim.cs.uiuc.edu/grace/.

[10] Thomas Granlund, Bo Granbom, and Nils Olsson. Soft error rate increase for new generations of SRAMs. *IEEE Trans. on Nuclear Science*, 50(6), Dec 2003.

[11] P. Hazucha and C. Svensson. Impact of CMOS technology scaling on the atmospheric neutron soft error rate. *IEEE Trans. on Nuclear Science*, 47(6):2586–2594, 2000.

[12] Hewlett Packard, http://www.hp.com. *HP iPAQ h5555 Series - System Specifications*.

[13] ITU-T. H.263 Draft: Video Coding for Low Bitrate Communication. Draft ITU-T recommendation H.263, ITU-T, May 1996.

[14] M. Kim, N. Dutt, N. Venkatasubramanian, and C. Talcott. xTune: Online verifiable cross-layer adaptation for distributed real-time embedded systems. *ACM SIGBED Review: Special Issue on the RTSS Forum on Deeply Embedded Real-Time Computing*, 5(1), Jan 2008.

[15] M. Kim, H. Oh, N. Dutt, A. Nicolau, and N. Venkatasubramanian. PBPAIR: An energy-efficient error-resilient encoding using probability based power aware intra refresh. *ACM SIGMOBILE Mobile Computing and Communications Review*, 10(3):58–69, July 2006.

[16] Kyoungwoo Lee, Minyoung Kim, Nikil Dutt, and Nalini Venkatasubramanian. Error exploiting video encoder to extend energy/QoS tradeoffs for mobile embedded systems. In *6th IFIP Working Conference on Distributed and Parallel Embedded Systems (DIPES)*, Sep. 2008, to appear.

[17] Kyoungwoo Lee, Aviral Shrivastava, Ilya Issenin, Nikil Dutt, and Nalini Venkatasubramanian. Mitigating soft error failures for multimedia applications by selective data protection. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, pages 411–420, Oct 2006.

[18] Jin-Fu Li and Yu-Jane Huang. An error detection and correction scheme for RAMs with partial-write function. In *IEEE International Workshop on Memory Technology, Design and Testing (MTDT)*, pages 115–120, 2005.

[19] Lin Li, Vijay Degalahal, N. Vijaykrishnan, Mahmut Kandemir, and Mary Jane Irwin. Soft error and energy consumption interactions: A data cache perspective. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 132–137, Aug 2004.

[20] Ritesh Mastipuram and Edwin C. Wee. *Soft Errors' Impact on System Reliability*. http://www.edn.com/article/CA454636, Sep 2004.

[21] S. Mohapatra, R. Cornea, H. Oh, K. Lee, M. Kim, N. Dutt, R. Gupta, A. Nicolau, S. Shukla, and N. Venkatasubramanian. A cross-layer approach for power-performance optimization in distributed mobile systems. In *Next Generation Software Program in conjunction with IPDPS*, page 218.1, April 2005.

[22] Shivajit Mohapatra, Radu Cornea, Nikil Dutt, Alex Nicolau, and Nalini Venkatasubramanian. Integrated power management for video streaming to mobile handheld devices. In *Proceedings of the 11th annual ACM international conference on Multimedia*, pages 582–591, 2003.

[23] MPEG. http://www.cselt.it/mpeg/.

[24] Shubhendu S. Mukherjee, Joel Emer, Tryggve Fossum, and Steven K. Reinhardt. Cache scrubbing in microprocessors: Myth or necessity? In *PRDC'04*, 2004.

[25] A. K. Nieuwland, S. Jasarevic, and G. Jerin. Combinational logic soft error analysis and protection. In *IOLTS06*, 2006.

[26] Richard Phelan. Addressing soft errors in arm core-based designs. Technical report, ARM, 2003.

[27] D. K. Pradhan. *Fault-Tolerant Computer System Design*. Prentice Hall, 1996. ISBN 0-1305-7887-8.

[28] Qi Qu, Yong Pel, J.W. Modestino, Xusheng Tian, and Bin Wang. Cross-layer design for delay-constrained error-resilient video communications over wireless ad-hoc networks. In *IEEE International Conference on Image Processing (ICIP)*, pages 720–723, Sep. 2005.

[29] P. Roche, G. Gasiot, K. Forbes, Oapos, V. Sullivan, and V. Ferlet. Comparisons of soft error rate for SRAMs in commercial SOI and bulk below the 130-nm technology node. *IEEE Trans. on Nuclear Science*, 50(6), Dec 2003.

[30] P. Shivakumar and N. Jouppi. CACTI 3.0: An Integrated Cache Timing, Power, and Area Model. In *WRL Technical Report 2001/2*, 2001.

[31] Aviral Shrivastava, Ilya Issenin, and Nikil Dutt. Compilation techniques for energy reduction in horizontally partitioned cache architectures. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, pages 90–96, 2005.

[32] Thomas Stockhammer. Robust system and cross-layer design for H.264/AVC-based wireless video applications. *EURASIP Journal on Applied Signal Processing*, 2006(1):270–270, 2006.

[33] Synopsys Inc., Mountain View, CA, USA. *Design Compiler Reference Manual*, 2001.

[34] Mihaela van der Schaar and Deepak S. Turaga. Cross-layer packetization and retransmission strategies for delay-sensitive wireless multimedia transmission. *IEEE Transactions on Multimedia*, 9(1):185–197, Jan. 2007.

[35] Mehmet C. Vuran and Ian F. Akyildiz. Cross-layer analysis of error control in wireless sensor networks. In *IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON)*, pages 585–594, Sep 2006.

[36] Yao Wang and Qin-Fan Zhu. Error control and concealment for video communication: A review. 86(5):974–997, May 1998.

[37] S. Worrall, A. Sadka, P. Sweeney, and A. Kondoz. Motion adaptive error resilient encoding for mpeg-4. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, pages 1389–1392, May 2001.

[38] F. Wrobel, J. M. Palau, M. C. Calvet, O. Bersillon, and H. Duarte. Simulation of nucleon-induced nuclear reactions in a simplified SRAM structure: Scaling effects on SEU and MBU cross sections. *IEEE Trans. on Nuclear Science*, 48(6):1946–1952, 2001.

[39] Jie Xu and Brian Randell. Roll-forward error recovery in embedded real-time systems. In *IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, page 414, 1996.

[40] Wanghong Yuan and Klara Nahrstedt. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. 37(5):149–163, Dec 2003.

[41] Wanghong Yuan and Klara Nahrstedt. Practical voltage scaling for mobile multimedia devices. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 924–931, 2004.